«دانش مدیریت»
سال دهم - شماره ۳۷ و ۳۸ - تابستان و پائیز ۱۳۷۶
ص ص ۱۸۷ - ۲۰۶

GENETIC ALGORITHM IN OPTIMAL PREVENTIVE PART REPLACEMENT FOR MINIMUM DOWNTIME MAINTENANCE PLANNING

By: NEZAMEDDIN FAGHIH (Ph. D.) & BABAK SOHRABI [1]

ABSTRACT

This paper innovates, within the context of preventive maintenance planning, the application of Genetic Algorithm, as a modern powerful optimization tool, in minimum downtime strategies, for optimal part replacement. Therefore, a brief account of Genetic Algorithm is given and a computer program is written to aid the solution of such problems by Genetic Algorithm. Then, the program is used to solve a numerical example, concerning an equipment with Gaussian failure characteristics. The results obtained by Genetic Algorithm and the ordinary methods are compared to show that exactly the same results can be approached by the application of

1- Shiraz University

Genetic Algorithm. Genetic Algorithm and maintenance planning, both being subjects of statistical nature, can be combined to provide a vast area of interesting research work and, hence, some guidelines are suggested for further research.

INTRODUCTION

Many analytical methods, for optimization problems, stop their search operations as they reach a local optimal solution and are unable to find global optimum answers. As an alternative approach to this major problem and in order to solve complex optimization problems, researchers have turned to use other techniques. Among the new methods, random algorithms, due to their simple structures and operations, also not limited to special types of goal functions, have been the most popular[1].

Genetic Algorithm constitutes a major class of random algorithms. It is based on biological aspects of natural evolution and was first introduced by J.H. Holland [2], [3]. Genetic Algorithm, being a statistical method for optimization and search, is not only a simple random search approach, but also leads to a good convergence[3]. Genetic Algorithm(GA) has been applied to many optimization problems in science and engineering, rendering satisfactory results[3], [4].

However, in the maintenance planning, optimal part replacement policies, mainly, employ statistical methods and the replacement function, based on the renewal theory, leading to the

minimization of a function such as downtime [5], [6].

Hence, Genetic Algorithm appears to be a suitable tool for optimal maintenance planning. In this paper, GA is applied to planning for optimal part replacement and preventive maintenance policies, to minimize downtime for equipment with normal failure rate. Nevertheless, a brief account of GA is, firstly, explained, through its basic operators.

BASIC OPERATORS IN GENETIC ALGORITHM

1) Coding:

Genetic algorithm uses coded shape of parameters and a usual way to code the parameters, is to convert them into binary digits; so an i..teger coding can be utilized [3], [4].

2) Chromosomes:

A string or array of bits, that is the coded shape of a probable answer to the optimization problem, is called a chromosome. In fact, bits in a string, have the same roles as gens have in nature. The initial values of the gens are chosen randomly [3], [4].

3) Population:

A group of chromosomes are called a population. One of the genetic algorithm abilities, is that it uses a population of chromosomes, and not limited points of probable answers in space or a single chromosome [3], [4].

4) Fitness Value:

Each string can be assigned a fitness value, based on how well

the corresponding problem solution meets some stated goal. Each individual›s probability of being reproduced, is proportional to the string›s fitness. Thus, the chromosome that has more fitness, would be used in the mating procces, more than the one that has less fitness [3], [4].

5) Cross Over Operator:

This operator uses two strings to reproduce two new answers and it can be single - point or multiple - point. Single - point operator mates two strings by joining the prefix of one string with the suffix of the other string, relative to the cross over point, which is selected randomly. This action creates two new strings which are called offsprings or children [3], [4].

Cross over operator selects the parents with some assigned probability, P_c [3], [4].

6) Mutation Operator:

Each binary digit has some small probability of being reversed, during the genetic recombination. This probability is usually shown by P_m and is called mutation probability [3], [4].

Now, that the primary operators of genetic algorithm have been introduced, one can focus on GA steps in the optimization problemes. In fact, the first step is the determination of parameters which would be used in the problem; then coding these parameters in a proper way in order to show them as binary strings. Based on the goal function, a fitness function for chromosomes is defined and an initial population is formed randomly. Hence, the fitness of each string can be

calculated and then optimization steps may be followed as depicted in Figure 1,[3].

In fact, the more fittness a chromosome may possess, the more is the chance that it is used in the reproduction operation; while chromosomes with less fitness may not be used in the reproduction. The simplest way to achieve this step, is to use the roulette wheel approach. In this model, the surface of the wheel is divided into parts; the number of parts being equal to the population size and the surface of each part being related to the fitness value used for choosing parents to mate for the reproduction process. The roulette wheel approach can be simulated on a computer, by following a random number generation scheme [3].

CONVERGENCE OF GENETIC ALGORITHM

An important point to raise is the convergence of Genetic Algorithm towards a global optimal answer. The research work undertook by Rudolph, in 1994, show [7] that GA convergence is possible and, while convergence towards the global optimum is not always a natural property of GA, but it can be achieved in some circumstances. In fact, using Markov chains model, it is shown that if in each step of GA reproduction, the fittest values were saved and transformed to the next step,with a probability of 1,GA would converge to global Optimum [7].

However the characteristics of GA may be summarized as follows [2],[7],[8],[9],[10]:

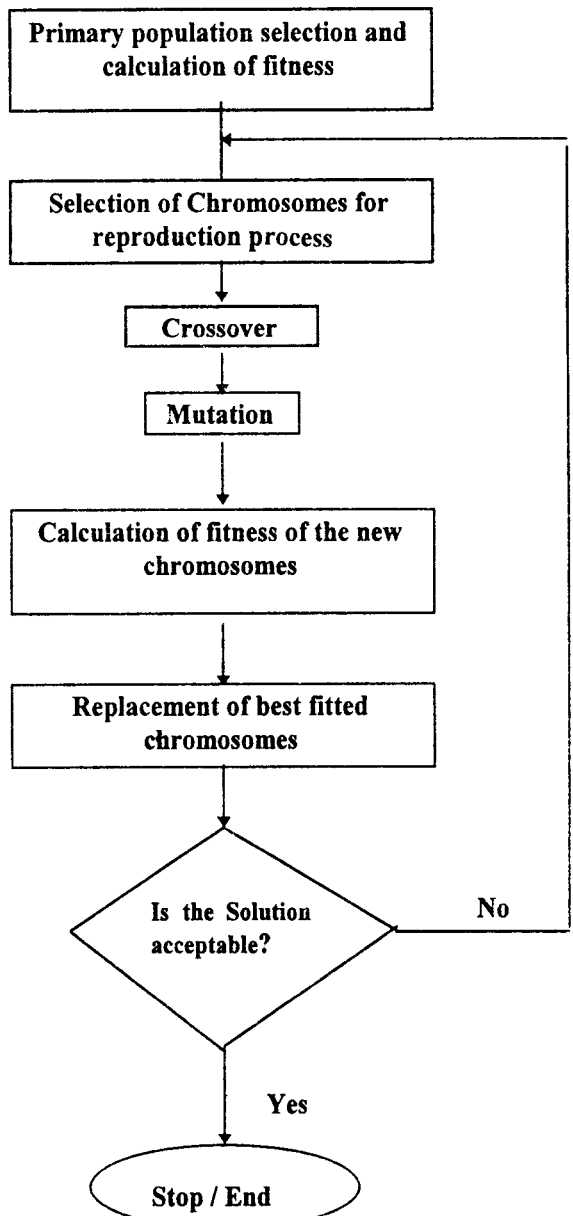

Figure 1: The optimization steps followed in Genetic Algorithm.

1) GA works with a population of chromosomes and is a parallel search method for optimum answer points with genetic operators. Information is transferred between points and because of that, the probability of trapping in a local optimum point is low.

2) By a good selection of new population in each iteration, GA converges to global optimum.

3) GA uses coded figures and parameters.

4) In GA, it is not necessary for the goal function to be differentiable, but GA only needs to compute the goal function in each point for achieving the global optimum and uses no other information.

5) GA, being a random search technique, is controlled by the fittness of chromosomes and transformation from one step to another is dependent upon fittness.GA uses genetic operators to search different parts of the solution space in order to achieve the global optimum.

GENETIC ALGORITHM IN OPTIMAL PART REPLACEMENT

In the maintenance planning, part replacement strategies,mainly, rely on the renewal function, which in its numerical form, appears as [5],[6]:

$$g(n) = \sum_{i=0}^{n-1} \{[1 + g(n-i-1)] \int_{iT}^{(i+1)T} p(t)dt\} \qquad (1)$$

The above iterative function, which in the replacement theory is

known as the replacement function, starts at g(0) = 0, due to the fact that g(nt.)=0, for n = 0, ie., there is no renewal at the time origine(t=0). There on, it renders the number of renewals or replacements as g(n), at discrete time intervals nT, for any number n, used as a multiple of an arbitrary constant time interval, T. The equipment is, however, supposed to fail according to some probability density function p(t).

In the optimal part replacement policies, the replacement function, given by equation (1), is used to formulate the required strategy. This is usually accomplished by minimizing or maximizing a desired objective function. As a major tendency, the optimality of part replacement policies are based on minimizing the total downtime, for preventive replacements and also required due to the random failures that may occur. Hence, the function to be minimized can be written as [5]:

$$D(T_r) = [\tau_r + g(T_r)\tau_f] / (T_r + \tau_r) \qquad (2)$$

In the above equation, the total downtime $D(T_r)$ is minimized with respect to the replacement time T_r. The parameters τ_r and τ_f are the times requied for a preventive replacement and a replacement forced due to failure, respectively. The value for $g(T_r)$ can also be computed from equation (1), for $T_r = nT$. In practice, the values of $g(T_r)$ are obtained, iteratively, from equation (1) and substituted in equation (2), to find the value of T_r which minimizes $D(T_r)$, as a numerical solution [5], [6].

As a numerical example, consider an equipment with a failure characteristic following the Gaussian probability distribution function with parameters, $\mu=7$ and $\delta= 2$ (weeks, say), as the mean and standard deviation, respectively. Suppose, also, that, $\tau_r = 0.0238$ and $\tau_f = 0.0476$ weeks. Hence, iterating equation (1), with T = 1 (week) and p (t) as a Gaussian probability density function with the given parameters, the values shown in Table 1 can be obtained. Then substituting these values for g (T_r), together with the given values of τ_r and τ_f, in equation (2), Table 2 can be computed. As it can be seen in Table 2, the minimum value of $D(T_r)$ occurs at T_r =5 (weeks); that is, the optimum preventive replacement period, for minimum downtime is 5 weeks.

n	0	1	2	3	4	5	6	7	8	9
g(n)	0	0.001	0.006	0.023	0.067	0.159	0.310	0.504	0.868	0.868

Table 1: Results of iterating equation (1) for Normal pdf with parameters μ =7, δ = 2 T = 1.

T_r	1	2	3	4	5	6	7	8	9
$D(T_r)$	0.0232	0.0119	0.0082	0.0067	0.0062	0.0064	0.0068	0.0071	0.0072

(minimum downtime)

Table 2: Numerical minimization of downtime $D(T_r)$ with respect to the preventive replacement period T_r, as computed from equation (2), by substituting the corresponding values.

Now, in order to apply GA to the problem of optimal

preventive part replacement, based on minimizing total downtime, according to equation (2), a computer program was written, in C (language). This program is presented in Appendix A. Here, instead of minimizing $D(T_r)$, the value of $1 / D(T_r)$ is maximized, which would rather simplify the task, as far as GA application is concerned

The above numerical example (worked out in table 2) was solved by employing the computer program in Appendix A. Some output samples are evidenced in Figures 2, 3, 4 and 5, which show that generations, finally settle at fixed values, rendering optimal solutions. The outputs read as x max = 5 and max = 160.24, which indicates T_r = 5 (weeks) and minimum value of $D(T_r)$ as $1 / 160.24 = 0.0062$. Therefore, the same results as observed from Table 2, are obtained by the application of Genetic Algorithm.

CONCLUSIONS

This paper has introduced the application of Genetic Algorithm to the problem of optimal maintenance planning and preventive part replacement for minimum total downtime of machinery. A computer program was developed and it was observed that exactly the same results as obtainable by ordinary methods, can be approached through the application of Genetic Algorithm.

The Genetic Algorithm, as a useful and powerful optimization method, employed by many researchers, provides a suitable approach for optimal maintenance policies, specially due to fact that both of them are subjects of statistical nature. It is hoped that the work

Population Report

Generation 3 Generation 4

#	string	x	fitness	#	Parents	xsite	string	x	fitness
0)	0100	4.00	149.18	0)	(4 , 5)	1	0100	4.00	149.18
1)	0110	6.00	156.40	1)	(4 , 5)	1	0100	4.00	149.18
2)	0100	4.00	149.18	2)	(2 , 8)	3	0100	4.00	149.18
3)	0100	4.00	149.18	3)	(2 , 8)	3	0110	6.00	156.40
4)	0100	4.00	149.18	4)	(1 , 8)	3	0110	6.00	156.40
5)	0100	4.00	149.18	5)	(1 , 8)	3	0100	4.00	149.18
6)	0110	6.00	156.40	6)	(0 , 9)	1	0100	4.00	149.18
7)	1000	8.00	140.40	7)	(0 , 9)	1	0110	6.00	156.40
8)	0100	4.00	149.18	8)	(3 , 2)	3	0100	4.00	149.18
9)	0100	4.00	149.18	9)	(3 , 2)	3	0100	4.00	149.18

Note: Generation 4 & Accumulated Statistics: xmax= 6 max=156.40 min=149.18

avg=151.34 sum=1513.44 nmutation=2 ncross=12

Figure 2: Output Sample for Generations 3 and 4.

Population Report

Generation 31 Generation 32

#	string	x	fitness	#	Parents	xsite	string	x	fitness
0)	0111	7.00	147.15	0)	(3 , 4)	0	0101	5.00	160.24
1)	0101	5.00	160.24	1)	(3 , 4)	0	0101	5.00	160.24
2)	0101	5.00	160.24	2)	(3 , 7)	0	0101	5.00	160.24
3)	0101	5.00	160.24	3)	(3 , 7)	0	1101	13.00	0.00
4)	0101	5.00	160.24	4)	(3 , 4)	2	0101	5.00	160.24
5)	0101	5.00	160.24	5)	(3 , 4)	2	0101	5.00	160.24
6)	0100	4.00	149.18	6)	(6 , 8)	1	0100	4.00	149.18
7)	0101	5.00	160.24	7)	(6 , 8)	1	1100	12.00	0.00
8)	0100	4.00	149.18	8)	(5 , 7)	3	0101	5.00	160.24
9)	1001	9.00	138.83	9)	(5 , 7)	3	0101	5.00	160.24

Note: Generation 32 & Accumulated Statistics: xmax= 5 max=160.24 min=0.00

avg=127.09 sum=1270.88 nmutation=46 ncross=104

Figure 3: Output Sample for Generations 31 and 32.

Population Report

Generation 45			Generation 46					
#	string	x	fitness	# Parents	xsite	string	x	fitness
0)	0001	1.00	42.90	0) (2 , 4)	0	0101	5.00	160.24
1)	0101	5.00	160.24	1) (2 , 4)	0	1101	13.00	0.00
2)	0101	5.00	160.24	2) (7 , 6)	2	0101	5.00	160.24
3)	0110	6.00	156.40	3) (7 , 6)	2	0101	5.00	160.24
4)	0101	5.00	160.24	4) (3 , 2)	0	0100	4.00	149.18
5)	0101	5.00	160.24	5) (3 , 2)	0	0111	7.00	147.15
6)	0101	5.00	160.24	6) (5 , 4)	1	0101	5.00	160.24
7)	0101	5.00	160.24	7) (5 , 4)	1	0101	5.00	160.24
8)	0100	4.00	149.18	8) (2 , 9)	2	0101	5.00	160.24
9)	0101	5.00	160.24	9) (2 , 9)	2	0101	5.00	160.24

Note: Generation 46 & Accumulated Statistics: xmax= 5 max=160.24 min=0.00
avg=141.80 sum=1418.03 nmutation=59 ncross=154

Figure 4: Output Samples for Generations 45 and 46.

Population Report

Generation 49				Generation 50				
#	string	x	fitness	# Parents	xsite	string	x	fitness
0)	0101	5.00	160.24	0) (1 , 8)	0	0101	5.00	160.24
1)	0101	5.00	160.24	1) (1 , 8)	0	0101	5.00	160.24
2)	0101	5.00	160.24	2) (8 , 4)	1	0101	5.00	160.24
3)	0101	5.00	160.24	3) (8 , 4)	1	0101	5.00	160.24
4)	0101	5.00	160.24	4) (8 , 6)	3	0101	5.00	160.24
5)	0101	5.00	160.24	5) (8 , 6)	3	0101	5.00	160.24
6)	0101	5.00	160.24	6) (5 , 2)	1	0101	5.00	160.24
7)	0001	1.00	42.90	7) (5 , 2)	1	0101	5.00	160.24
8)	0101	5.00	160.24	8) (0 , 6)	0	0101	5.00	160.24
9)	0101	5.00	160.24	9) (0 , 6)	0	0101	5.00	160.24

Note: Generation 50 & Accumulated Statistics: xmax= 5 max=160.24 min=160.24
avg=160.24 sum=1602.43 nmutation=64 ncross=167

Figure 5: Output Samples for Generations 49 and 50.

reported in this paper, paves the way for further work in this interesting area and applying GA to various maintenance planning strategies. As far as minimum downtime preventive part replacement is concerned, however, it is suggested that further research work may be undertaken on the application of GA, while considering the possibilities of eliminating the renewal theory and, hence, the replacement function.

REFERENCES

1 - Lasser, J.B., Baraiya, P.P. and Warland, J.(1987), "Simulated Annealing, Random Search, Multistart or SAD?", Systems & Control Letters, North - Holland, pp. 297 - 301.

2 - Holland, J.H.(1989), "Searching Nonlinear Functions for High Values", Applied Mathematics & Computation, 32, pp. 255 - 274.

3 - Lawrence, D.(1991), "Handbook of Genetic Algorithms", Reinhold Publishers.

4 - Goldberg, E.D.(1989), " Genetic Algorithms in Search, Optimization and Machine Learning", Addison - Wesley publishers.\

5 - Jardine, A.K.S. (1973), "Maintenance, Replacement and Reliability", Pitman publishers.

6 - Cox, D.R.(1962), " Renewal Theory", Methuen / Wiley.

7 - Rudolph, G. (1994), " Convergence Analysis of Canonical Genetic Algorithms", IEEE Transactions on Neural Networks, Vol.5, No.1, pp. 96 - 101.

8 - Sriniras, M. and Patanaik, L.M.(1994), "Adaptive Probability of Crossover and Mutation in Genetic Algorithm", IEEE Transactions on Power Systems, Vol.24, No.4, pp. 656 - 666.

9 - Galar, R.(1989), "Evolutionary Search with Soft Selection", Biological Cybernetics, Springer Verlag, pp.357 - 364.

10- Karloy, F.P. (1993), "Genetic Algorithms for the Travelling Salesman Problem Based on a Heuristic Crossover Operation", Biological Cybernetics, Springer Verlag, pp. 539 - 546.

APPENDIX A

```
#include "stdlib.h"
#include "math.h"
#include "conio.h"
#include "stdio.h"
#include "dos.h"
#include "time.h"
#include "alloc.h"
#include "math.h"
#define maxpop 100
#define maxstr 30
#define pi 3.141592654

char **dim2(int row,int col,unsigned int size);
void free2(float **pa);
float integ(int a, int b, int mu, int sig);
double randm(void);

struct individ
{
 int chrom[maxstr];
 float x;
 float fitness;
 int parent1;
```

```
 int parent2;
 int xsite;
};

 struct individ oldpopu[maxpop];
 struct individ newpopu[maxpop];

 int popsize, lchrom, gen, maxgen, nmutate, ncross, jrand=-1, interv,
xmax;
 float pcross, pmutate, sumfit, avg, max, min,**g,tof,tor;
 double oldrand[55]={0};
 FILE *fpt;
 FILE *fl;

main( )
{
  int j;

   gen=0;
   ininormal( );
   initialize( );
   gotoxy(30,12);
   printf("PLEASE WAITE");

   for (gen=1; gen<=maxgen; gen++)
   {

     generation( );
     statistics(newpopu);
     report(gen);
     for(j=0;j<maxpop;j++)
     {
         oldpopu[j]=newpopu[j];
     }

   }
  gotoxy(28,13);
 printf("END OF SIMULATION");
```

```
 fclose(fpt);
 fclose(f1);
 free2(g);
}


ininormal(void)
{
        int miu, sigma, n, i;
        char name1[15], name2[15];

clrscr( );
gotoxy(1,9);
printf("        ------------------------------------------------\n");
printf("         A Genetic Algorithm Process Simulator -
GENETICA\n");
printf("                        Babak Sohrabi \n");
printf("                   Industerial Management Dept.      \n");
printf("                    Shiraz University 1996\n");
printf("        ------------------------------------------------\n");
getch( );

  clrscr( );
 printf("All of the results of this simulation are gathered in two
 files\n");
 printf("the first one contains generations and the second one
contains\n");
  printf("the MAX and AVG of each generation (for plotting).\n");
  printf("\nEnter output files name:\n");
  scanf("%s %s",name1,name2);
  fpt=fopen(name1,"w");
  f1=fopen(name2,"w");

  clrscr( );
  printf("\n********** Parameters Of The desired process
**********\n");
  printf("\nEnter mean and St.Dev. of the process (Weeks):\n");
  scanf("%d %d",&miu,&sigma);
```

```
 printf("Enter the interval of simulation (Weeks):\n");
 scanf("%d",&interv);
 printf("Enter tor and tof (Hours):\n");
 scanf("%f %f",&tor,&tof);

fprintf(fpt,"-------------------------------------------------\n");
fprintf(fpt,"A Genetic Algorithm Process Simulator - GENETICA\n");
fprintf(fpt,"              Babak Sohrabi \n");
fprintf(fpt,"        Industerial Management Dept.      \n");
fprintf(fpt,"          Shiraz University 1996\n");
fprintf(fpt,"-------------------------------------------------\n");

 skip(3);
 fprintf(fpt,"\n********* Parameters Of The Desired Process
*********\n");
 fprintf(fpt,"\nMean = %d weeks  St.Dev. = %d weeks\n",miu,sigma);
 fprintf(fpt,"\nInterval of simulation = %d weeks\n",interv);
 fprintf(fpt,"\ntor = %.2f hours  tof = %.2f hours\n\n",tor,tof);
 fprintf(fpt,"\nThe results of the RENEWAL FUNCTION for
NORMAL PDF\n");
 fprintf(fpt,"with Mean = %d and St.Dev. = %d\n",miu,sigma);
 fprintf(fpt,"-------------------------------------------------\n");

 tof = tof/(24*7);
 tor = tor/(24*7);

  g=(float **)dim2(1,interv+1,sizeof(float));
  for(n=0; n<=interv; n++){ g[0][n]=0; }

 for(n=1; n<=interv; n++)
 {
 for(i=0; i<n; i++)
 {
  g[0][n] = g[0][n] + (1 + g[0][n-i-1]) * (integ(i,i+1,miu,sigma));
 }
fprintf(fpt,"g[%d] = %.3f\n",n,g[0][n]);

 }

}
```

```
float integ(int a, int b, int mu, int sig)
{

  float h, fi, la, it, t=a, ev=0, od=0;
  int j;

  h = fabs((b-a))/100;
  fi = (1/(sig*sqrt(2*pi))) * exp(-(a-mu)*(a-mu)/(2*sig*sig)) * 0.5;
  la = (1/(sig*sqrt(2*pi))) * exp(-(b-mu)*(b-mu)/(2*sig*sig)) * 0.5;

  for(j=1; j<=100-3; j=j+2)
  {

    t = t + h;
    od = od + (1/(sig*sqrt(2*pi))) * exp(-(t-mu)*(t-mu)/(2*sig*sig));

    t = t + h;
    ev = ev + (1/(sig*sqrt(2*pi))) * exp(-(t-mu)*(t-mu)/(2*sig*sig));

  }
  t = t + h;
  od = od + (1/(sig*sqrt(2*pi))) * exp(-(t-mu)*(t-mu)/(2*sig*sig));

  it=(fi + la + 4*od + 2*ev)*h/3;

  return(it);
}


char **dim2(int row,int col,unsigned int size)

{ int i;
  char **prow,*pdata;
pdata=(char *)calloc(row*col,size);
if(pdata==(char *)NULL){
  printf("Interval of simulation is too long. Reduce it.");
  exit(1); }
prow=(char **)malloc(row*sizeof(char *));
 if(prow==(char **)NULL){
```

```
      printf("Interval of simulation is too long. Reduce it.");
          exit(1); }
     for(i=0;i<row;i++){
      prow[i]=pdata;
        pdata += size*col;
            }
        return prow;
                  }
void free2(pa)
 float **pa;
 {   free(*pa);
     free(pa);   }


double objfunc(double x)
{
/* const float coef=1073741823;
 const int n=10;

 double y;
  y=pow(x/coef,n);
  return(y);*/

 double y;

 if(x > interv)
 {
  y = 0;
 }

 else
 {
  y = (x + tor) / (tor + g[0][x]*tof);
 }
 return(y);
}
 double decode(int crom[maxstr], int lbits)
 {
  int j;
```

```
 float accum=0, powof2=1;
  for(j=0; j<lbits; j++)
  {
   if(crom[j]==1){ accum=accum+powof2; }
   powof2=powof2 * 2;
  }
  return(accum);
}

statistics(struct individ popu[maxpop])
{
 int j;

  sumfit = popu[0].fitness;
  min    = popu[0].fitness;
  max    = popu[0].fitness;
  xmax   = popu[0].x;

  for(j=1;j<popsize;j++)
  {
   sumfit = sumfit + popu[j].fitness;
   if(popu[j].fitness>max){ max=popu[j].fitness; xmax=popu[j].x; }
   if(popu[j].fitness<min){ min=popu[j].fitness; }
  }

  avg = sumfit/popsize;
}

void initdata(void)
{

clrscr( );
printf("******** Genetic Algorithm Data Entry and Initialization
********\n");
printf("\n");
printf("Enter population size -------> "); scanf("%d",&popsize);
printf("Enter chromosome lenght -----> "); scanf("%d",&lchrom);
printf("Enter max. generations ------> "); scanf("%d",&maxgen);
printf("Enter crossover probability -> "); scanf("%f",&pcross);
```

```
printf("Enter mutation probability --> "); scanf("%f",&pmutate);
sleep(1); clrscr( );
randomiz( );
sleep(1); clrscr( );
nmutate=0;
ncross=0;
}

initreport(void)
{
skip(3);
```